次世代ロボット知能化技術開発プロジェクト

手先拘束下でのマニピュレーション
知能モジュール
仕様書・取扱説明書

2011年3月30日

国立大学法人　東北大学

株式会社　パイケーク

更新履歴

改定日付	
2009/12/14	新規作成
2010/05/18	5 章にモデルファイルに関する記述を追加
2010/05/31	コンパイル手順の変更に対応
2011/03/30	OpenRTM1.0 対応

目次

１．はじめに

手先拘束下でのマニピュレーション知能モジュールは、以下の機能を実装した RT コンポーネントです。

- 拘束されている空間では力制御を行う
- 自由な空間ではコンプライアンス制御を行う
- 選択行列の値によって、コンプライアンス制御と力制御の切り替えを行う

図 1 は手先拘束下でのマニピュレーション知能モジュールと、他のモジュールとの接続関係を表した概要です。

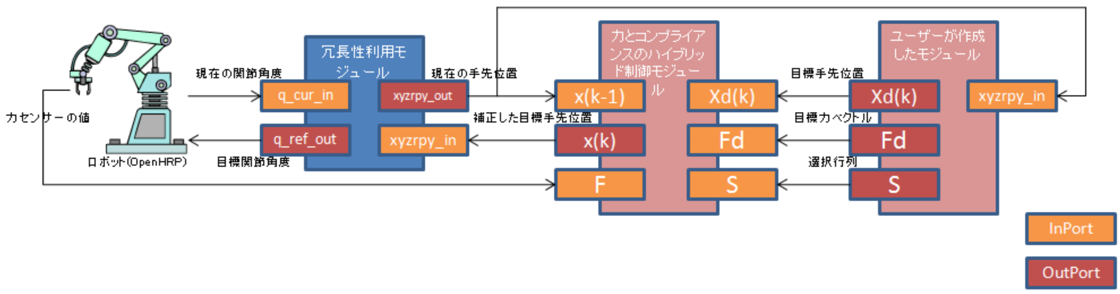


図 1 手先拘束下でのマニピュレーション知能モジュール概要

本モジュールは，以下の文献を参考に作成しました。

永田寅臣，渡辺桂吾，佐藤和也，泉清高，末廣利範，
オープンアーキテクチャ型の産業用ロボットのための位置指令型インピーダンス制御， 精密工学会誌，
Vol. 64, No. 4, pp. 552--556, 1998.

２．手先拘束下でのマニピュレーション知能モジュール

２．１．データ入出力ポート

力とコンプライアンスのハイブリッド制御モジュールが提供するデータ入出力ポートは以下の通りです。

ポート名称	入出力	概要
x(k-1)	入力	ロボットアームの現在の手先位置
xd(k)	入力	ロボットアームの手先の目標位置
x(k)	出力	補正したロボットアームの手先の目標位置
F	入力	ロボットアームの力センサの値
Fd	入力	目標力ベクトル
S	入力	選択行列。x, y, z, roll, pitch, yaw 方向ごとに、1: コンプライアンス制御, 0: 力制御, を選択する

２．２．コンフィギュレーションインターフェース

力とコンプライアンスのハイブリッド制御モジュールで設定可能なコンフィギュレーションインターフェースは以下の通りです。

コンフィグレーション名称	データ型	概要
Δt	double	サンプリングタイム
Bd_0～5	double	ロボットの粘性
Kd_0～5	double	ロボットの剛性
Kf_0～5	double	力制御ゲイン

３．サンプルシミュレーション

手先拘束下でのマニピュレーション知能モジュールの利用例として、OpenHRP3 に付属の GrxUI を利用したサンプルシミュレーションについて説明します。

３．１．接続例

図 2 はサンプルシミュレーションを実行する際のモジュールの接続構成です。

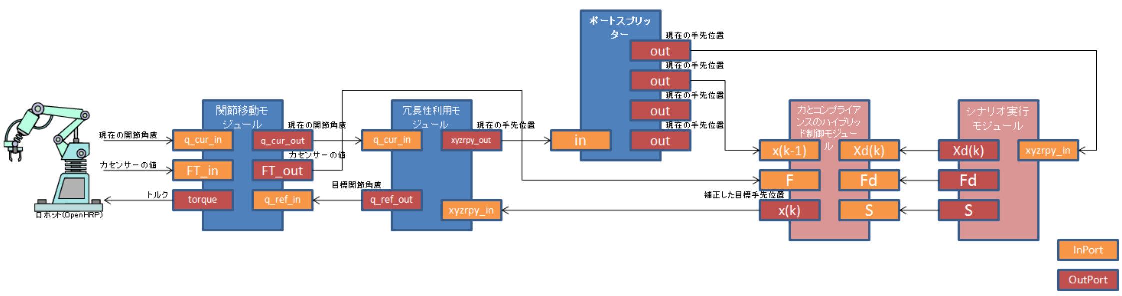


図 2 接続例

図 3 は RT System Editor 上で関節移動モジュール、冗長性利用モジュール、手先拘束下でのマニピュレーション知能モジュール、ポートスプリッダーモジュール、シナリオ実行モジュールを接続した例です。

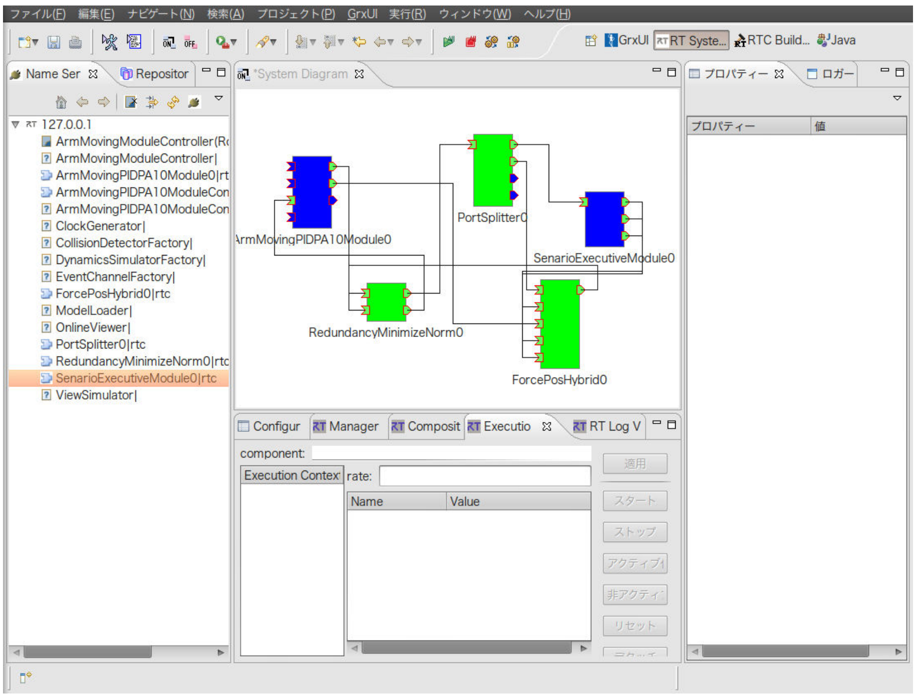


図 3 接続例

３．２．関節移動モジュール（PD 制御バージョン）

シミュレータと冗長性利用モジュールは入出力ポートのインターフェース仕様が異なるため直接接続することはできません。そのため、関節移動モジュールはシミュレータと冗長性利用モジュールとのブリッジの役目を果たしています。

関節移動モジュールは現在のロボットアームの関節角度(q_cur_in)から、冗長性利用モジュールから設定された目標関節角度(q_ref_in)へ徐々に近づける動作を行います。PD 制御によって必要なトルクを計算し、シミュレータ側へ出力します。

また、シミュレータから出力される力センサーの値を、力とコンプライアンスのハイブリッド制御モジュールに渡しています。

データ入出力ポートについては以下の通りです。

ポート名称	入出力	概要
q_cur_in	入力	ロボットアームの現在の関節角度
q_cur_out	出力	ロボットアームの現在の関節角度。内容は q_cur_in と同じ。他のモジュールから参照するために使用
q_ref_in	入力	ロボットアームに設定する目標関節角度
FT_in	入力	ロボットアームの力センサーの値
FT_out	出力	ロボットアームの力センサーの値。内容は FT_in と同じ。他のモジュールから参照するために使用
torque	出力	ロボットアームに設定するトルク

コンフィギュレーションインターフェースについては以下の通りです。

コンフィグレーション名称	データ型	概要
factor_d	double	微分ゲインの重み
factor_p	double	比例ゲインの重み
gain_d_0～6	double	微分ゲイン
gain_p_0～6	double	比例ゲイン
torque_limmiter	double	トルクの上限

３．３．冗長性利用モジュール

冗長性利用モジュールの詳細については、冗長性利用モジュールに付属のドキュメントを参照してください。

３．４．ポートスプリッダーモジュール

ポートスプリッダーモジュールは 1 つのポートに入力される値を、 4 つポートにそのまま出力するモジュールです。

冗長性利用モジュールの出力である現在の手先位置は、手先拘束下でのマニピュレーション知能モジュールと、ユーザーが作成したプログラムの両方に入力する必要があるため用意しています。

データ入出力ポートについては以下の通りです。

ポート名称	入出力	概要
in	入力	ロボットアームの現在の手先位置
out	出力	ロボットアームの現在の手先位置

３．４．デモシナリオ実行モジュール

デモシナリオ実行モジュールは、デモ用のシナリオファイル(pos.dat)を読み込んで、目標手先位置 Xd(k)、目標力ベクトル Fd、選択行列 S を手先拘束下でのマニピュレーション知能モジュールに出力するモジュールです。

デモシナリオ実行モジュールが読み込み可能な pos.dat は、1 行につき、次の書式でデータを格納します。各数値はスペースまたは tab で区切りで記述します。

xyzrpyのそれぞれ値(6要素) 選択行列の対角成分(6要素) 目標力ベクトル(6要素)

データ入出力ポートについては以下の通りです。

ポート名称	入出力	概要
xyzrpy_in	入力	現在の手先位置
Xd(k)	出力	目標手先位置
Fd	出力	目標力ベクトル
S	出力	選択行列

４．インストール

４．１．必要環境

各モジュールは下記の環境で動作確認を行っています。

- Ubuntu 10.04 Desktop Edition
  - http://www.ubuntu.com/
- OpenRTM-aist-1.0.0-RELEASE
  - http://www.openrtm.org/openrtm/node/849
- OpenRTM-aist-Python-1.0.0-RELEASE
  - http://www.openrtm.org/openrtm/node/932
- rtshell
  - http://www.openrtm.org/openrtm/ja/node/869
- OpenHRP3 Ver.3.1.0-Release
  - http://www.openrtp.jp/openhrp3/jp
- Ruby 1.8.7
  - http://ruby-lang.org/ja/
- log4cxx
  - http://logging.apache.org/log4cxx/

Ubuntu 10.04 Desktop Edition をインストール後、必ず

```
$ apt-get update & apt-get dist-upgrade
```

のコマンドを実行し、ライブラリ・プログラムなどを最新状態にしておきます。

OpenRTM 環境のインストールは次の手順で行ってください。

```
$ wget
http://openrtp.jp/openrtm/svn/OpenRTM-aist/trunk/OpenRTM-aist/build/pkg_install1
00_ubuntu.sh
$ sudo sh pkg_install100_ubuntu.sh
$ wget
http://www.openrtm.org/pub/OpenRTM-aist/python/install_scripts/pkg_install_pytho
n_ubuntu.sh
$ sudo sh pkg_install_python_ubuntu.sh
$ wget http://www.openrtm.org/pub/OpenRTM-aist/tools/1.0.0/rtctree-3.0.0.tar.gz
$ tar xvfz rtctree-3.0.0.tar.gz
$ cd rtctree-3.0.0
$ sudo python setup.py install
$ cd ..
$ wget
http://www.openrtm.org/pub/OpenRTM-aist/tools/1.0.0/rtsprofile-2.0.0.tar.gz
```

```
$ tar xvfz rtsprofile-2.0.0.tar.gz
$ cd rtsprofile-2.0.0
$ sudo python setup.py install
$ cd ..
$ wget http://www.openrtm.org/pub/OpenRTM-aist/tools/1.0.0/rtshell-3.0.0.tar.gz
$ tar xvfz rtshell-3.0.0.tar.gz
$ cd rtshell-3.0.0
$ sudo python setup.py install
$ cd ..
```

OpenHRP3 Ver.3.1.0-Release のインストールは次の手順で行います。

```
$ sudo vi /etc/apt/source.list
  次の行を追加
  deb http://archive.canonical.com/ubuntu lucid partner

$ sudo apt-get update
$ sudo apt-get install openhrp3.1

$ sudo update-alternatives --config java
  ※java-6-sunが選択されていることを確認

$ wget
http://www.openrtp.jp/openhrp3/download/eclipse342_hrpdependencies_linux_ja_2010
0602.tar.gz
$ tar xvfz eclipse342_hrpdependencies_linux_ja_20100602.tar.gz
$ mv eclipse eclipse342_hrpdependencies_linux_ja_20100602
$ vi eclipse342_hrpdependencies_linux_ja_20100602/eclipse.ini

  次の修正を行う
  --- eclipse.ini.org  2011-03-02
                         18:08:42.275122261 +0900
  +++ eclipse.ini  2011-03-02
                         18:08:50.792122115 +0900
  @@ -7,5 +7,5 @@
   --launcher.XXMaxPermSize
   256m
   -vmargs
  --Xms64m
  --Xmx512m
  +-Xms256m
  +-Xmx1024m

$ cp /usr/share/OpenHRP-3.1/java/plugins/*.jar
eclipse342_hrpdependencies_linux_ja_20100602/plugins/
$ sudo mv eclipse342_hrpdependencies_linux_ja_20100602 /usr/local/
$ cd ~

$ mkdir bin
```

```
$ vi ~/bin/eclipse.sh

  次の内容を記述

  #!/bin/sh
  export GDK_NATIVE_WINDOWS=1
  /usr/local/eclipse342_hrpdependencies_linux_ja_20100602/eclipse -vmargs
-Dorg.eclipse.swt.browser.XULRunnerPath=/usr/lib/xulrunner-1.9.2.13/xulrunner
-clean
※起動時には~/bin/eclipse.shを使ってEclipseを起動すること。
```

現状の OpenHRP3.1.0-Release には pkg-config の設定にパスが正しく設定されていない問題があるため、OpenHRP3 関係のソースコードをコンパイルを行うことができない不具合があります。正しくコンパイルを行うことができるように /usr/lib/pkgconfig/openhrp3.1.pc ファイルを次のように修正します。

```
prefix=/usr/local
  ↓
prefix=/usr
```

コンパイルに使用する Ruby と log4cxx は、次の手順でインストールを行います。

```
$ sudo apt-get install ruby ruby-dev
$ sudo apt-get install liblog4cxx10 liblog4cxx10-dev
```

４．２．コンパイル

各モジュールは*.tar.gz ファイルを取得・展開した後、make コマンドを使用してコンパイルを行います。ファイル名の中の????の部分にはモジュールのリリース日付が入ります。

以後 force_pos_hybrid_module-201?????.tar.gz を展開したファイルをホームディレクトリの下の work ディレクトリに配置している前提で説明します。

```
$ cd ~
$ tar xvfz force_pos_hybrid_module-201?????.tar.gz
$ mv force_pos_hybrid_module-201???? work
$ cd work

$ ls
Makefile*
force_pos_hybrid_module-201?????.tar.gz
PA10_owa-201?????.tar.gz                 hybrid-demo.xml
README.txt
port_splitter-20110329.tar.gz
arm_moving_pid_pa10_module-201?????.tar.gz
redundancy_minimize_norm-201?????.tar.gz
build.rb*
senario_executive_module-201?????.tar.gz

$ make extract
  ※各モジュールのtar.gzファイルを展開します。

$ make
  ※各モジュールのコンパイルを行います。
```

５．サンプルシミュレーションの実行手順

同梱しているサンプルシミュレーションでは、アームが壁にぶつかった後、 50N の力で壁を押しつづける様子をシミュレートします。 GrxUI の力センサの値が壁に対してほぼ 50N で押し返され続ける様子がグラフに表示されます。

サンプルシミュレーションを実行する際は、次の手順で操作を行います。

1. GrxUI を起動します。
2. work ディレクトリ内で"make setup"を実行し､デモに必要なモジュールを起動します。
3. work ディレクトリ内にあるデモ用プロジェクトファイルである hybrid-demo.xml を GrxUI で開きます。
4. GrxUI の"Start Simulation"ボタンを押すと、シミュレーションが開始されます

- ロボットアームの手先が壁に接触した後、50N の力で壁を押しながら壁をなぞる動作を行います。

シミュレーションを開始すると、アームは緑色の壁に接触し、接触した後壁をなぞる動作を行います。壁に接触し壁をなぞる動作を行っている最中は、アームは力制御により常に 50N で壁を押し続けます｡図 4 の GrxUI の右側は手先の圧力センサの値を表示しています。グラフをみると壁に接触した後は、常に約-50N の力が計測されていることが確認できます。

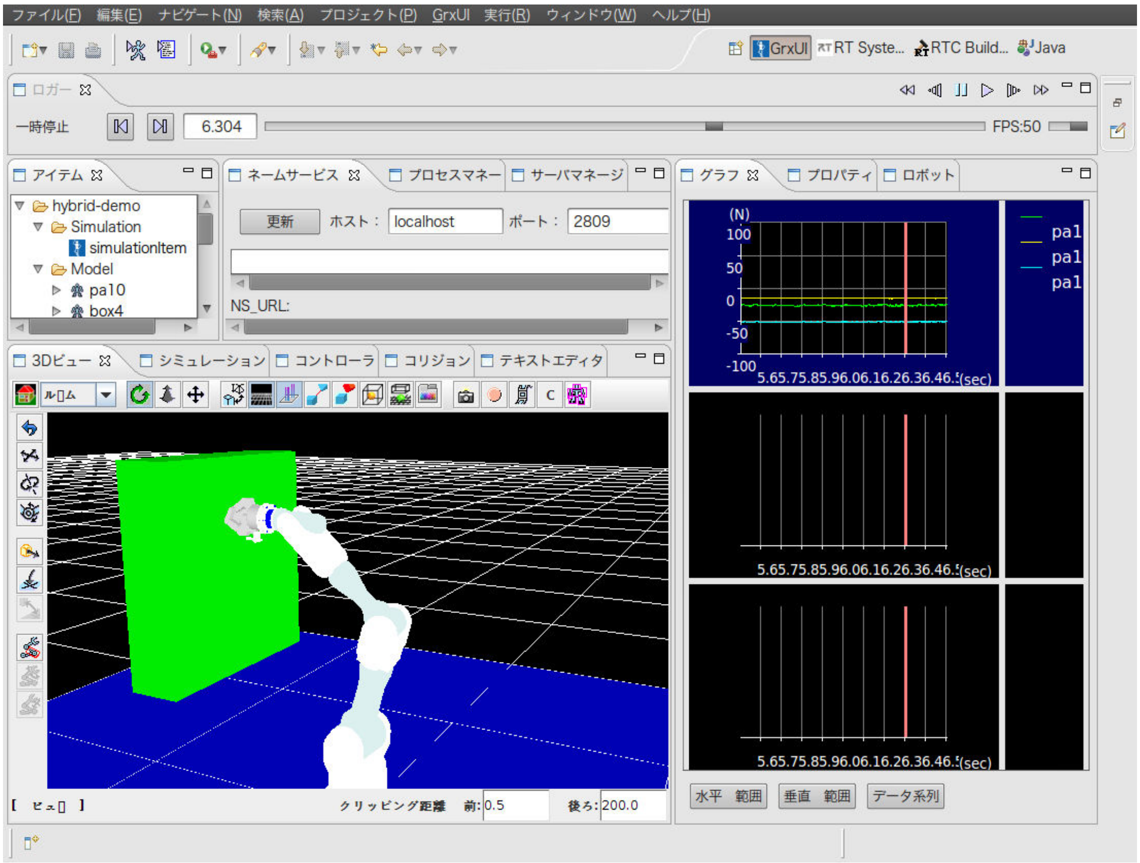


図 4 シミュレーション中の GrxUI の様子

各モジュールを一度に終了する場合は、work ディレクトリ内で "make shutdown"を実行してください。